AUTOMATED
MISSILE AIM POINT SELECTION
TECHNOLOGY
Final Report

Contract #N00014-92-C-0087

January 1995

Submitted to:

Dr. William Miceli
Office of Naval Research - 1264
800 North Quincy Street
Arlington, VA 22217-5000

Submitted by:

Gary Whitten, Principal Investigator
Martin Marietta Labs
1450 South Rolling Road
Baltimore, MD 21227
Phone: (410) 204-2686
Fax: (410) 204-2100
email: whitten@mml.mmc.cor

INTRODUCTION

Martin Marietta Labs - Baltimore (MMLB) has developed missile aimpoint algorithms that address the requirements of the ONR AMAPS program by maintaining an awareness of the dominant issues. To be of practical use to the AMAPS program, Automatic Target Recognition (ATR) algorithms must be effective and reliable over a wide range of weather, lighting, visibility, obscuration, and terrain conditions in cluttered and noisy environments. There should be minimal restrictions on type of target, target orientation, and viewing direction. Further, effective performance must be achieved within the power, speed, size, and cost constraints dictated by real world operational conditions.

The model-based approach to ATR (MBATR) has shown much promise in achieving high levels of target discrimination of reduced target signatures in highly cluttered environments (as required for target identification sufficient for distinguishing between two similar targets in the same class i.e., friendly tank from enemy tank in battlefield conditions). The model based approach derives target information from target and sensor models, so the target information is not limited by an unwieldy training set of images and the tedious and unreliable training task is not required. The model based approach achieves robustness of recognition by finding an optimum match between target models and image data through a powerful hypothesis prediction and verication scheme, (see figure 1). However, the powerful MBATR techniques come at the expense of potentialy

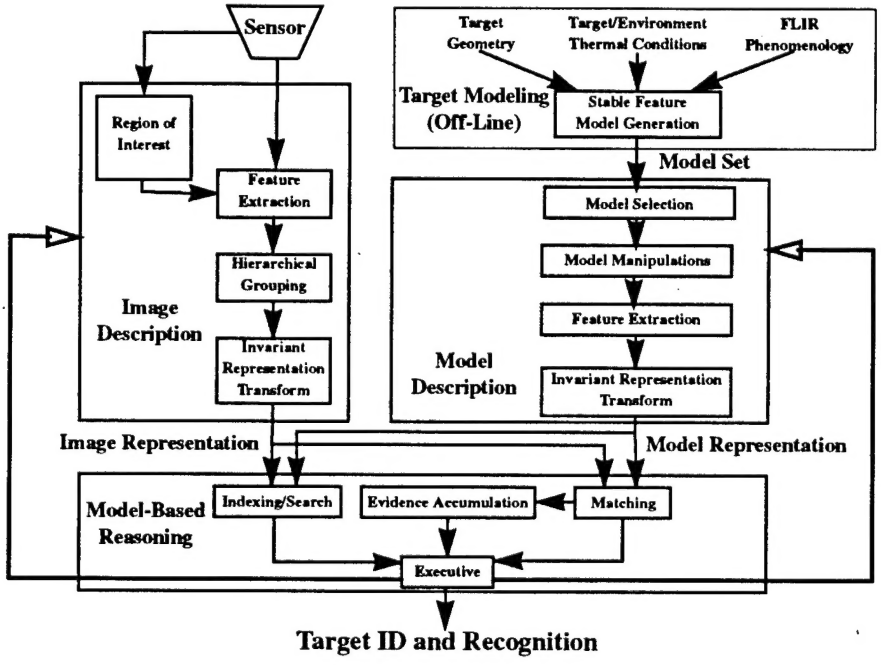


Figure 1. Model-Based ATR System

high computational complexity due to the required search through the space of model instances for the best match with observed data. An undirected and unconstrained search performed directly in the image domain can not be implemented efficiently enough to be of practical use in the AMAPS scenario.

To address this problem for the AMAPS program, MMLB has developed highly efficient directed search strategies using its Vertex Space invariant representation (a representation that is stable over some useful range of viewing conditions, such as geometry and thermal properties, in the case of IR imagery) that reduces and constrains the required search space such that the full power of MBATR can be realized in a practical implementation. Further, we have developed techniques to extract stable local features and their attributes from real imagery. We use vertex (or corner) features and the local properties of vertex size and orientation, which are mapped to Vertex Space, to direct the search process. To achieve robustness, our feature extraction techniques do not require explicit edge extraction. Rather, our feature extraction process is based on local image orientation properties, so we avoid the problems of edge analysis such as disontinuous edges and complex contour tracking. Use of local features can accommodate incomplete and/or degraded target signatures (due to occlusion, poor visiblity and other adverse conditions) and avoid the pitfalls of region segmentation.

Vertex Space decreases the size and dimensionality of the search space required for model to data matching by exploiting the invariant properties of vertices extracted from images. An extracted vertex (intersecting straight lines or contour segments of maximum curvature) can be essentialy described by its location and orientation in the image, as well as the size of the angle subtended by its intersecting tangent lines. Although the spatial location of a vertex is highly sensitive to image formation details, its orientation and angle size are invariant to many of these details but can still characterize essential object structure. Therefore, our vertex representation emphasizes the two invariant properties, angular orientation and size. A vertex in image space maps to a point in Vertex Space as determined by its angular size and orientation, independent of its spatial location in the image (see figure 2).

A Vertex Space mapping is insensitive to changes of scale, translation, and rotation within the image plane so it serves as a viewpoint insensitive characteristic signature for the object. Since Vertex Space is affected only by changes in vertex size or orientation, Vertex Space is clearly invariant to image translations and changes in scale. Image plane rotation results only in a constant shift in the orientation dimension of Vertex Space. But, since orientation is defined with respect to an arbitrary axis, only relative orientation is meaningful (see figure 3).The insensitivity of Vertex Space to many of the details of two dimensional imaging geometry (a useful approximation for high altitude imaging and objects confined to a plane) results in enormous recognition simplification. For two dimensional recognition, Vertex Space provides an invariant signature for targets and the

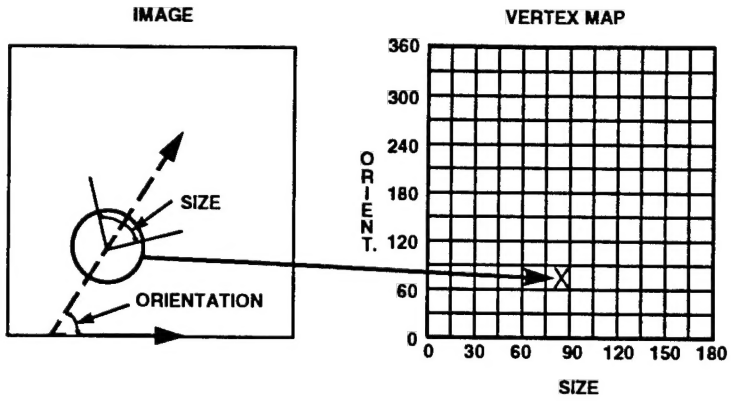


Figure 2) MAPPING TO VERTEX SPACE

matching process reduces to simple template matching in Vertex Space with no need for hypothesis generation, since all Vertex Space representations are equivalent (see Figure 4). In Vertex Space, a single template is sufficient, while if performed using correlation in the image domain, hundreds of templates would be necessary to cover the required range of scale and orientation. In addition, the approach is robust with respect to corrupted or occluded information since each vertex contributes independently to matching confidence, unlike techniques that depend on reliable segmentation of signature silhouettes. The usefulness of Vertex Space invariance extends to the more general three dimensional recognition problem (see figure 5).

Three dimensional imaging geometry is characterized by six degrees of freedom,

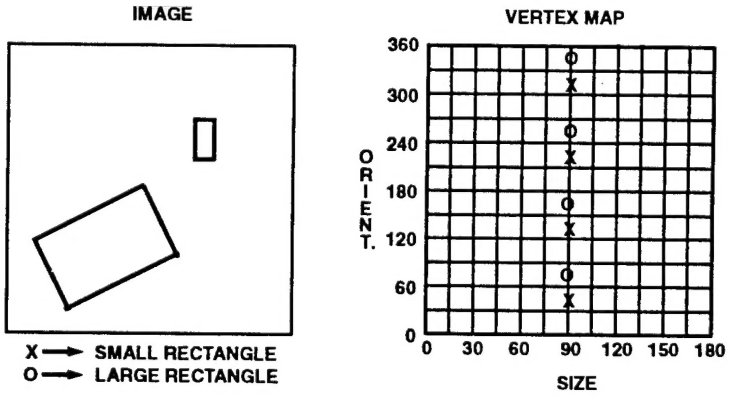
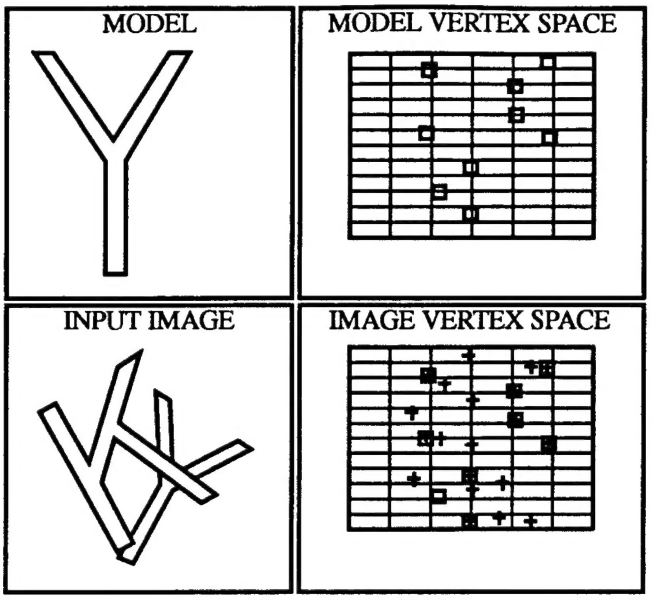


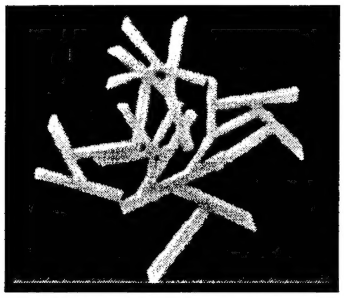
Figure 3) INSENSITIVITY OF VERTEX SPACE TO TRANSLATION, SCALE AND 2D ROTATION



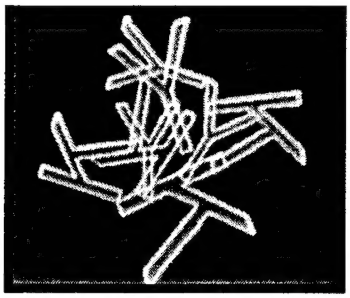
1 TEMPLATE
for
VERTEX SPACE MATCHING
vs.
250 TEMPLATES
for
IMAGE DOMAIN MATCHING

(based on 200% scale variation
1 template per
10% scale variation
1 template per
10 degrees rotation)

Vertex space object recognition process. For 2D objects, recognition is reduced to simple template matching. (Note the occlusion of the center "Y" vertex.)



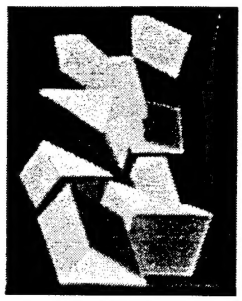
Input image consisting of multiple, occluding 2-D objects ("Y" and "K") in various sizes and orientations.



Recognized objects are highlighted by superposing projections of the model objects.

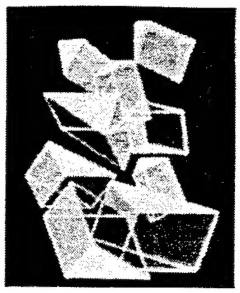
Figure 4) 2D OBJECT RECOGNITION USING VERTEX SPACE

three associated with translation and three associated with orientation. Without loss of generality, we can define a composite rotation about three orthogonal axes, such that the last rotation is about the optical axis. The advantage of such a coordinate system is the optical axis rotation is equivalent to rotation within the image plane and produces only a simple constant offset in Vertex Space, which does not affect target signatures. Of the six spatial degrees of freedom, only the two associated with rotations about axes orthogonal to the optical axis have a meaningful affect on a Vertex Space representation of an object. The corresponding reduction of the search space greatly simplifies the search process for three-dimensional model to data matching. Matching in Vertex Space provides a complete specification of viewing geometry (by determining correspondence among three



GEOMETRIC DEGREES of FREEDOM:

VERTEX SPACE - 2 vs. IMAGE DOMAIN - 6



Multiple occluding 3D solid objects in various orientations and sizes.

Recognized objects are highlighted by superposing projections of the model objects.

Figure 5) 3D VERTEX SPACE OBJECT RECOGNITION

or more model and image points) so that final matching can be performed by mapping the entire model object into the image. Vertex Space is used for highly efficient preliminary matching, or prescreening, and only when a match in Vertex Space is found is the more costly image space matching performed. Like the two dimensional recognition approach, described above, three dimensional model based recognition using Vertex Space greatly simplifies the initial matching process and provides robustness with respect to occluded or corrupted data.

In the application of Vertex Space to 3D ATR, the 3D geometric and sensor properties of a target model are automatically and systematically encoded into a set of compact 2D Vertex Space representations, where each representation is derived from a single synthetic view of the object (using appropriate sensor models), which solves many of the problems commonly associated with the efficient creation and manipulation of model databases for complex targets. Features from sensor data are analyzed in a similar way and their invariant properties index into the model database and direct the search process, yielding hypotheses consisting of candidate representations, their associated 2D views and potential correspondences of model features with sensor image features. The generated hypotheses are quickly verified or rejected by performing efficient matching between candidate sensor and model featues, which provides a powerful prediction, verification mechanism that is highly effective for avoiding false target declarations.

TECHNICAL APPROACH

Search Strategy

Our emphasis in the development of techniques to perform model to image matching is efficiency with respect to clutter and robustness with respect to missing or corrupted image data. To perform direct matching using feature locations is a problem of

$O((N_i*N_m)^3)$ complexity, where N_i is the number of observed image features and N_m is the number of model features. Typically, N_i could be 1000 and N_m could be 100, resulting in a problem of immense computational complexity. An important issue, closely related to computational efficiency, is the matching mechanism. Matching 2D image data directly to 3D models is a difficult problem since it requires detecting all potential 3D features that could project to a 2D feature, which, if done rigorously, requires examining regions, in all possible directions, about every surface point. Even if these points could be effectively found directly in the 3D domain, many more features than necessacry would have to be considered simultaneously, since only a small subset is typically visible from any single view (i.e., all edges of rectangles that aproximate a cylinder must be considered as possible matches with a straight line in an image, even though, typically, only two will ever be visible at the same time). In contrast, it is straightforward to match 2D critical points extracted from images directly with 2D projections of the model. For this reason, we generate a database of views of a given 3D target model and perform matching with the set of views rather than the model itself.

To avoid the complexity of exhaustive image to model feature matching, we use feature attributes and the invariance of Vertex Space to direct a multi-stage search process. The search strategy is hierarchical where viewing geometry and feature grouping is constrained incrementaly so that each search level performs a small subset of the whole search, greatly reducing the combinatoric complexity, and subsequently reducing the data that must be considered at the next stage (notice the data flow structure in figure 13).

The first stage of the search exploits the invariant properties of Vertex Space to efficiently index into a model database of views using a hash table indexing approach which is an excellent match with Vertex Space since hashing techniques require a binary array representation, that Vertex Space naturally provides. The result of Vertex Space indexing is candidate model views and image to model feature correspondence. The hypotheses generated by indexing are then checked for orientational consistency, which determines an offset angle of image rotation. Surviviing candidate features are then evaluated to detemine positional consisentcy, which is the first stage of the search process that uses any feature location data. Positional consistency is based on predicted orientations of lines defined by feature pairs, so scale information is not needed. The surviving features are finaly checked for scale consistency. The result is a correspondence between image features and model features as well as a full approximate description of the viewing geometry.

The model to image feature correspondence can then be used to refine the viewing geometry which, in turn, can be used to map the complete target model into the acquired image for final correpondence determination. The final stage involving analysis of the complete target model is costly, but is only performed when there is high confidence that

a target is present as a result of the efficient feature matching search. The constraints enforced by the feature matching process ensure that false alarms will be unlikely. It can be readily seen that, while our approach is efficient, each step of the search process is directed by the results of previous steps, it can also accomodate missing features since the search process is valid for any subset of features. We describe the details of our search process in the next section.

Search Approach: Curvature Directed Search Using Vertex Space

Our mulitstage search process begins with indexing. Indexing is the initial prediction of target, viewpoint, and correspondendce between model and sensor features based on the most significant features extracted from the sensor image data. Indexing generates hypotheses to be investigated. Specificaly, viewpoint hypotheses are generated based on candidate correspondences between image and target features. In addition to viewpoint, other conditions affecting target signatures are considered, i.e., thermal properties via the appropriate sensor models.

The off-line preperation of an indexing database requires extensive manipulation and analysis of the target model. However, real target models can be highly complex making manipulation difficult. To avoid the diffiiculties resulting from 3D target model feature analysis, we produce a compact database of Vertex Space representations derived from the systematic generation of synthetic views of the target model, using the appropriate sensor model. Each 2D image generated is associated with a different 3D view of the target object. The different views that contribute to the model database can be associated with points uniformly sampled at some specified resolution from a sphere surrounding the model object where each point defines a location from which to view the object.

If not for the invariance of Vertex Space, we would also need to consider points of varying distance along rays centered on the object (spheres of different radius) and, at each point, different rotations about the optical axis. However, the translation (including scale) and rotational invariance of Vertex Space requires that only a single view from each point on the tessellated sphere be generated.

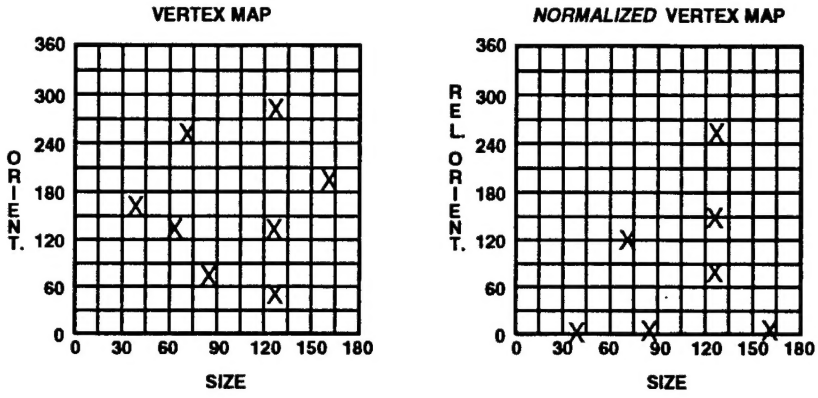
Analytically, the viewing geometry transformation that maps a point, whose location is determined by the 3D position vector, $\mathbf{x}$, to the transformed point, $\mathbf{x}'$, is described by the 3D transformation

$$\mathbf{x}' = \mathbf{R}_z(\theta_z)(\mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y))\,\mathbf{x} + \mathbf{s}$$

where $\mathbf{R}_k(\theta_k)$ represents a rotation of θ_k about axis $\mathbf{k}$, and $\mathbf{s}$ is a translation vector. Vertex Space is invariant with respect to $\mathbf{s}$ and will only be shifted by a constant amount, θ_z, along

Fig. *6)* NORMALIZED VERTEX SPACE

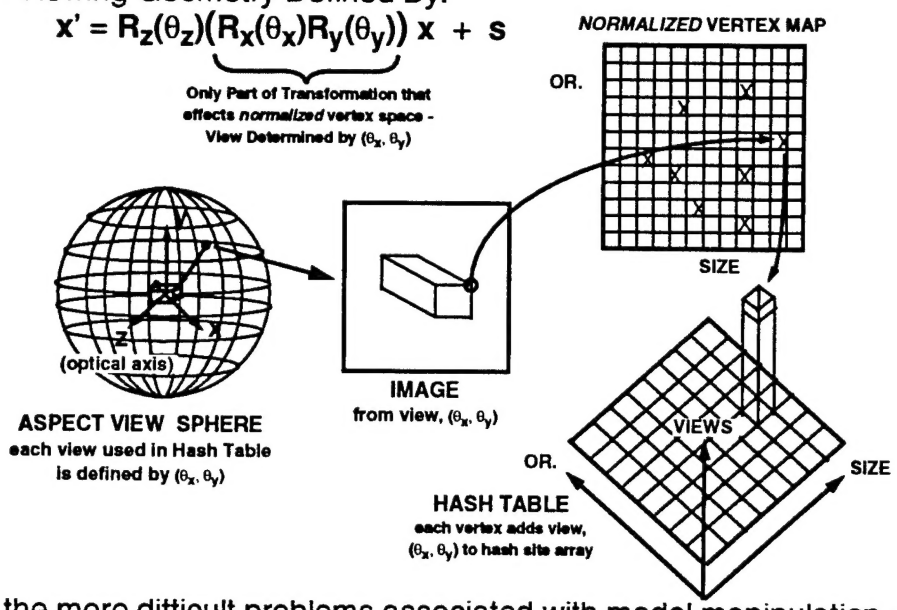
record ***RELATIVE*** pairwise difference orientations of vertices in same size bin instead of absolute orientations



the orientation axis. To achieve complete invariance with respect to $R_z(\theta_z)$ (rotation within the image plane), we normalize Vertex Space by mapping all pairs of vertices within the same size bin to Normalized Vertex Space bins based on the average size of the vertex pair and a *relative* orientation, found by taking the difference between the two vertex orientations in the pair (see Figure 6). Normalized Vertex Space removes the dependence on orientation offset by considering only orientation differences. Forming vertex pairs results in an increase in the number of features (N -> N * (N-1), where N is the number of vertices in a size bin) that must be analyzed, but relative orientation information can be used to realize much more distinctive Vertex Space signatures than size information alone, resulting in more robust indexing, which increases the efficiency of the subsequent search stages. We form pairs based on vertices with similar size attributes since many targets of interest have symmetric features. When symmetric features undergo viewing transformations, they tend to maintain size similarities. Forming pairs from all possible feature pairs would result in too many features. In dense data domains (highly complex targets, high resolution imagery or high clutter) it may be more appropriate to use standard (unnormalized) Vertex Space and perform inexing with explicit quantized orientation offsets on single vertices to avoid forming too many vertex pairs. The details of determining which approach to use, single vertices and absolute orientation in standard Vertex Space or vertex pairs and relative orientation in Normalized Vertex Space, will be developed in the complexity analysis below.

We proceed by synthetically generating each of the views defined by the tessellated sphere. We map the resulting synthetic image to *normalized* Vertex Space and, for each vertex, we enter the current view index number into a hash table at the corresponding normalized Vertex Space site so that the hash table accumulates all the view indices that are associated with each observed vertex in Vertex Space (see figure 7). As mentioned earlier, by using an aspect hashing graph to represent geometric object information, we

Fig. 7) BUILDING HASH TABLE ASPECT MODEL DATABASE

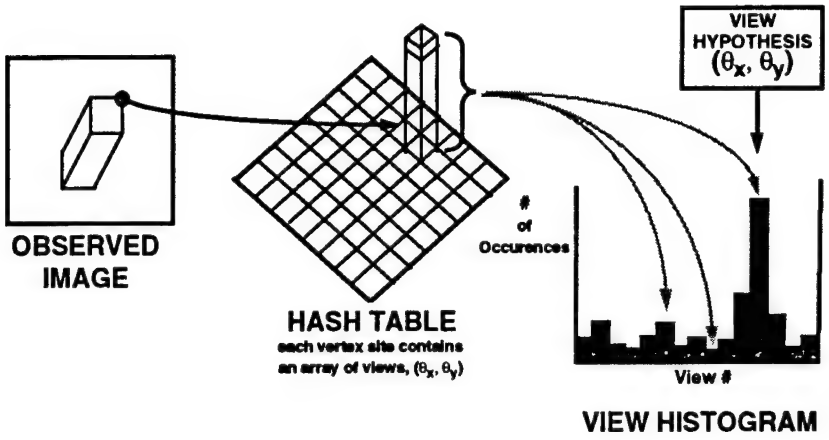


avoid one of the more difficult problems associated with model manipulation, which is the determination, from the model, of visible features. Rather, we generate all required views explicitly, off-line, so the visibility of edges is determined via the synthetic image generated for each view used to build the hash table. We have automated the hash table generation process so the hash tables can be generated for any object that has a model in our system and both view angle and hash table angle resolutions can be specified. We perform hash table generation, model manipulation, synthetic image generation, algorithm design, implementation and evaluation on our MB-ATR system testbed which uses the C, Unix, X Windows and Silicon Graphics GL graphics standards.

We employ a 2 stage indexing process using the hash table to determine viewpoint and model to image vertex correspondence. In the first stage the θ_y, θ_x parameters are found by mapping the observed image to normalized Vertex Space. For each vertex mapped, the corresponding hash table site is accessed and the set of views stored at the site are distributed into a viewpoint histogram. The resulting peaks indicate candidate views (see figure 8).

The θ_y, θ_x view parameters found from the first stage of the indexing process are used to identify a candidate model Vertex Space map, which is compared to the image Vertex Space map. Sets of potentially corresponding model and image vertices define an index into θ_z values, the final parameter required to completely specify object orientation. Specifically, we recover θ_z and the model to image vertex correspondence from the second stage of indexing. The hash table indexing provides a (θ_y, θ_x) hypothesis and associated model Vertex Space representation which is compared to the observed image Vertex Space representation, which is *not* normalized (normalization is only required for

Fig. 8) 2 STAGE INDEXING - First Stage:
Using **HASH TABLE** to find View Paramaters, (θ_x, θ_y)



the first stage of indexing). A correspondence between a model and image Vertex Space entry defines an offset angle, θ_z. The resulting value is accumulated in a θ_z histogram and any peaks indicate candidate θ_z values (see Figure 9). Once a model Vertex Space map is identified, additional constraints can be derived from positional properties of the object view, which can verify or reject the working (θ_y, θ_x, θ_z) hypothesis and further constrain the viewing geometry and model to data vertex correspondence, as described below.

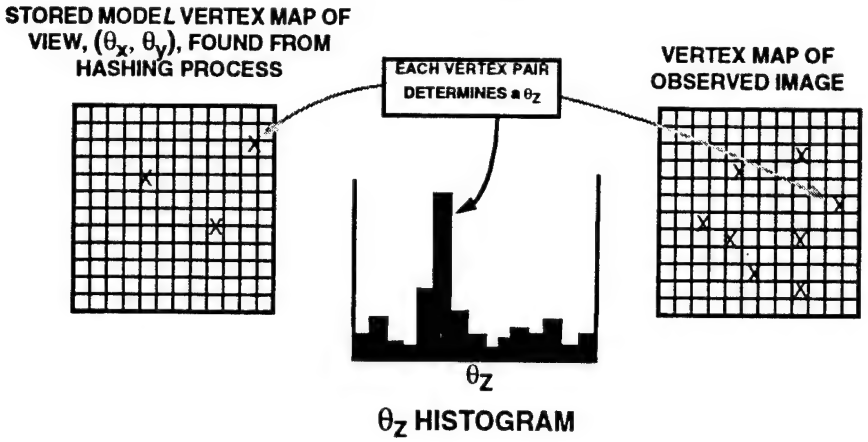
Our indexing scheme requires mapping continuous data into discrete grids so it is subject to quantization error, where vertex entries straddle bin boundaries, which can seriously degrade performance. The quantization error is aggravated by smaller bin sizes, which introduce more bin boundaries, thereby increasing the likelihood that vertex entries will fall near bin boundaries. To solve this problem, we distribute hash table entries about the single bin determined by the quantized values. A table entry is placed in the discrete bin it normally falls into, and is also placed in its (3) nearest neigbor bins with weights determined by the distance from the desired continous value of the table entry to the center of each bin (see Figure 10). This ensures that angular values that straddle bin borders still register, even if they should cross over the bin boundary, which allows for increasing the bin resolution arbitrarily without concern for adverse quantization effects. This approach has effectively removed all problems associated with discrete quantization error.

The result of the indexing process described above is a candidate (θ_y, θ_x) view, and θ_z optical axis rotation and candidate correpsondence between model and image features, which typically identifies a small subset of all image features. The (θ_y, θ_x, θ_z) hypothesis is equivalent to defining a properly rotated (in the image plane) 2D view of the

Fig. 9) 2 STAGE INDEXING - SecondStage:

Find θ_Z (Orientation Offset Angle)

- process determines model to observed vertex correspondence
- vertex correpondence used to find all viewing geometry



model object. It is still necessary to determine the translation and scale transformation (a subset of the general affine 2D transformation) required to align the hypothesized view with the observed data. Determination of the required 2D transformation requires matching an image pair of features with a corresponding pair of model features (assuming each feature has a well defined point location). Verification of the hypothesized 2D transformation requires at least one further model to image feature pair correspondce. So the geometric matching problem, after the indexing stage, reduces to a pair wise search.

We employ simple geometric constrains to determine and verify the 2D transformation necessary (along with (θ_y, θ_x, θ_z)) for fully defining the viewing geometry. While the indexing process deals excusively with local properties of individual vertices (i.e., angle size and orientation), the remaining search uses vertex positions. Without knowing scale,

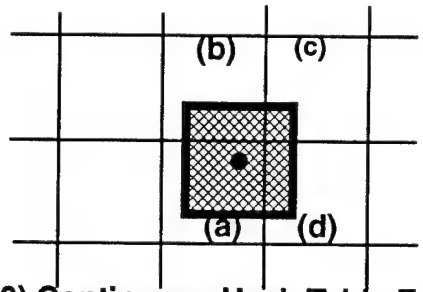


Figure 10) Continuous Hash Table Entries

An area the size of a hash bin (indicated by cross-hatching) is centered around the continuous hash table value (indicated by the dark circle). A value is stored in each hash bin equal to the size of the overlap area. In this case, bin a would receive the highest entry value, followed, repectively, by b, d and c.

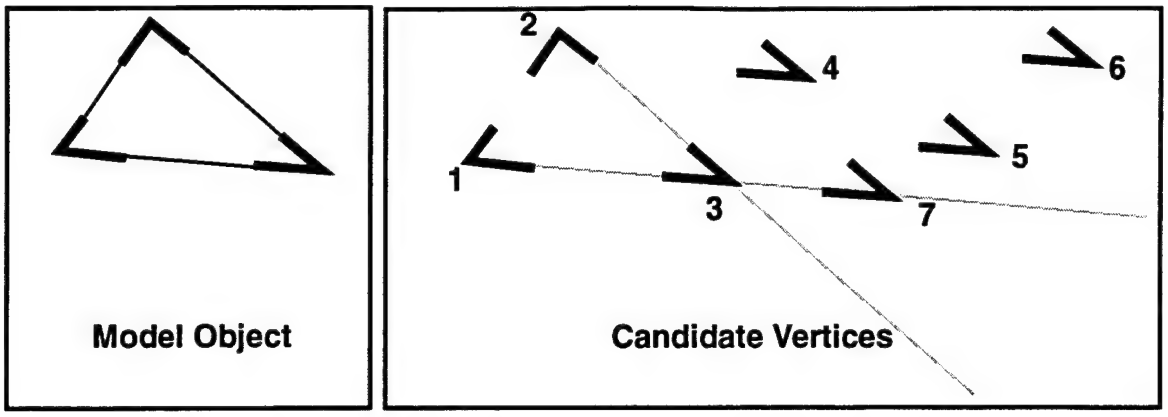


Figure 11) Positional Consistency in θ_z Indexing

Based on the Vertex Map derived from (θ_y,θ_x) indexing alone, image vertices 1 and 2 and any of 3, 4, 5, 6, or 7 could be potential matches with the model object. However, if we incorporate simple positional information that could be stored with the vertex data, the ambiguaties could be readily resolved. Specifically, the angle $\overline{13}$ makes with $\overline{23}$ would reduce the possibilities for the third vertex to either 3 or 7. When the additional constraint imposed by $\overline{32}$ with $\overline{12}$ is considered, only one choice (#3) remains. No scale information is required.

we can inter, from the model, the orientation of a line connecting any two corresponding image vertices. Therefore, one candidate image vertex constrains each other candidate image vertex to a specific line. Once an additional candidate vertex is found to lie on the line specified by the first, the two constitute a consistent candidate vertex pair that completely constrain the locations of all other candidate vertices and define the viewing geometry (by determining the previously unknown 2D translation and scale transformations - see Figure 11).The viewing geometry specified by such a base pair can then be verified by the remaining candidate image vertices.

To perform this pairwise search and verification process efficiently, we do a breadth first determination of all possible pair consistencies and store the binary ("1" for consistent, "0" for not consistent) results in a consistency matric, **C**. The remainder of the search can then be highly directed by geometric consistency constraints. The total number of potential mutually consistent candidate vertices, T_m, associated with a single reference vertex is the sum of all "1" entries in all matrix elements associated with the reference vertex. The search starts with the single reference vertex having the highest number of potential matches, as indicated by T_m. All "1" entries in the base reference row are consistent with the reference vertex. Each of these entries can be used to reference another row. All entries common to these two rows are mutually consistent and satisfy all geometric constraints. Therefore, we select a first reference row based on the T_m values. For each entry in this row, the corresponding row is checked for common entries, which must be mutually consistent. This process is repeated, for descending T_m values until the

minimum required number of verified image vertices are found or until there are no more rows with T_m values equal to or greater than the minimum required match number (see figure 12).

Once the image set of vertices that matches with the model object has been identified and the viewpoint determined, the model object can be mapped into the image for a final image domain correspondence determination, fully verifying or rejecting the Vertex Space hypothesis.The matching vertices that contributed to the view determination can be used to find a least squares solution that will precisely refine the viewing geometry estimate found in the matching process described above.

Our overall search strategy decomposes the computationaly intensive end-to-end model to image matching search into smaller search components. Each of these is far simpler and computationaly more tractable than the global search as well as further constraining and determining the imaging geometry and establishing candidate image to model vertex correspondences. These search components are hierarchicaly related such that each level further reduces the amount of image data that must be dealt with by the next level (as represented by the line widths in figure 13) in a "divide and conquer" strategy.

Feature Based Matching Complexity Analysis

To analyze the complexity of our model to image feature matching search strategy, we

Figure 12) CONSISTENCY MATRIX, C - Drives Pairwise Search:

- **C** stores binary results of all pairwise consistency checks
- look for vertices that are consistent with a common vertex (look for matrix rows with common elements, candidate rows determined by elements in prior rows)
- choose reference index based on most consistent pairings, T_m

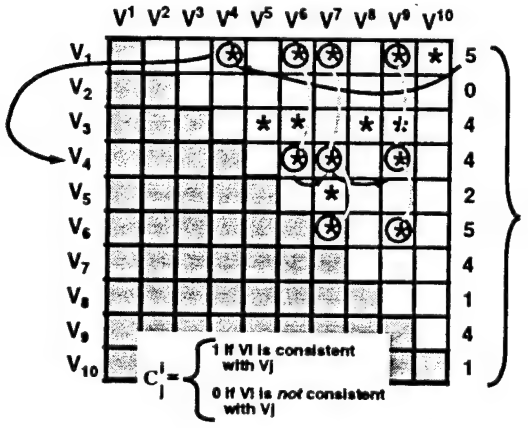
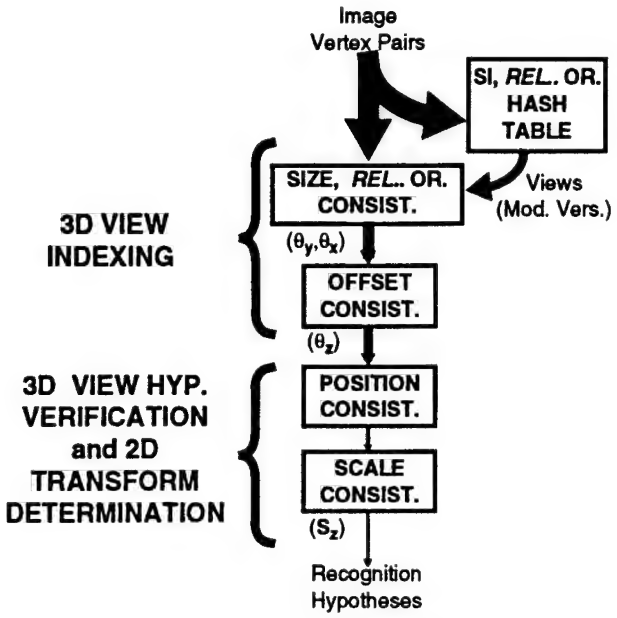


Fig. 13) HIERARCHICAL, DIRECTED SEARCH STRATEGY



- **Search strategy decomposes search into simpler search components**
- **Each component implements simpler search and further reduces data to be analyzed by next stage**

make the distinction between the first module of the search, indexing into the 3D view and the second module, which verifies (or rejects) the 3D view hypothesis and determines the appropriate 2D transformation which includes both scale and translation information (see figure 13). In the following analysis, VS is Vertex Space, and *we make the assumption that vertices tend to be uniformly distributed over Vertex Space*, which of course is not strictly true, but, from subjective observation of clutter experiments, it appears to be a reasonable approximation.

N_i is the number of Image Vertices, M_i is the average Image VS bin Population

N_m is the number of Model Vertices, (average for all views) M_m is the average Model VS bin Population

N_{si} is the number of VS size bins, N_{or} is the number of VS orientation bin

(Therefore, assuming uniform distribution,

$$M_i = N_i / (N_{si} * N_{or}) \quad , \quad M_m = N_m / (N_{si} * N_{or}))$$

N_v is the number of (θ_x, θ_y) views, N_o is the number of (θ_z) offset values

From our experiments, we have found the following values to be effective:

$N_{si} \Rightarrow 18$, $N_{or} \Rightarrow 36$ (from 10 degree quantization)

$N_v \Rightarrow$ 178 (15 degree step size) ; $N_o \Rightarrow 24$ (15 degree quantization)

3d View Indexing

We use three different versions of 3D view indexing: 1) no orientation, 2) relative orientation and 3) absolute orientation. The efficiency of each is dependent on the amount of image clutter, complexity of the target, and resolution of Vertex Space. The first version uses size information alone, which produces many view hypotheses since size alone is much less discriminating than size and orientation together, however, the use of feature pairs, which increases the number of feautures to be analyzed, can be avoided. Image pairs are formed to exploit relative orientation which is encoded in the Vertex Space hash table. The resulting view hypotheses are stronger due to the combination of size and orientation discrimination, however, there are more features that must be analyzed. Finaly, we encode absolute orientation into the hash table and use single features, which requires that we explicity shift the orientation of Vertex Space to accomodate the N_o orientation offsets.

- No Orientation (no) Information

For no orientation information, the indexing process involves accessing the Vertex Space Hash Table (size only) with the size value of each image vertex. At each Hash Table site there are M_v^{no} view entries, so there are a total of $M_v^{no} * N_i$ opertions to be performed. M_v^{no} can be approximated by multiplying the probability that any given model vertex will map to a given Hash Table size bin by the number of views, which is $(N_m / N_{si}) * N_v$ (where (N_m / N_{si}) is constrained to less than or equal to 1). So the complexity is

$$C1^{no}_1 = N_i * (N_m / N_{si}) * N_v$$

- Relative Orientation (ro) Information

To use relative orientation information, the N_i vertices form vertex pairs with other vertices that fall within the same size bin, each of which defines a relative orientation value used to access the Vertex Space Hash Table. There are N_i^2/N_{si} such pairs. At each Hash Table site there are M_v^{r0} view entries, which result in $(M_v^{r0} * N_i^2) / N_{si}$ operations. M_v^{r0}, the probable number of views at a given Hash Table site, can be approximated by the probability that a given view will have at least one model vertex pair entry at a given site times the number of views. So M_v^{r0} is $N_m^2 / (N_{si}^2 N_{or})$ and

$$C1^{ro}_1 = ((N_i^2 N_m^2) / (N_{si}^3 N_{or})) * N_v$$

- Orientation Offset Consistency

View indexing with no orientation information or with relative orientation information defers dealing with absolute orientation values. Absolute orientation is found by

determining orientation offset consistency, that is, each candidate model to image vertex match defines an orientation offset which, in turn, defines absolute orientations. For consistency, the same offset orientation must apply to all model to image vertex associations in the same hypothesis, so these groupings are all placed in the appropriate offset bin for further analysis. This operation, which follows the first indexing stage described above requires $N_m * M_i$ operations. Which is the number of model vertices (N_m) times the number of image vertices that could be associated with each model vertex (M_i). Each model vertex (from a given view) is assigned to a Vertex Space bin and all image vertices that fall within the same bin are candidate matches for the model vertex. Using the uniform vertex distribution assumption, there are

$$M_i = N_i / (N_{si} N_{or})$$

image vertices in a given Vertex Space bin.

which we approximate with, $(N_i N_m) / (N_{si} N_{or})$. This term must be added to $C1^{no}$ and $C1^{ro}$ since it represents the second stage of the view indexing module, however, it is clearly negligable with respect to both $C1^{no}$ and $C1^{ro}$, so it can be ignored and we have

$$C1^{no} = (N_i N_m / N_{si}) N_v$$

and

$$C1^{ro} = ((N_i N_m)^2 / (N_{si}^3 N_{or})) N_v$$

- Absolute Orientation (ao) Information

View indexing using absolute orientation information involves accessing the Vertex Space Hash Table with each of the extracted image vertices for each of N_o offset values. Therefore, $N_i * M_v^{ao} * N_o$ operations are required, where M_v^{ao} is the probable number of views residing at any Hash Table element, approximated by $(N_m / (N_{si} N_{or})) * N_v$. The complexity for view indexing using absolute orientation is

$$C1^{ao} = ((N_i N_m) / (N_{si} N_{or})) N_o N_v$$

3D View Indexing Comparison

Since the resolution of orientation bins and orientation offset bins must be similar for consistency (although they do not need to be identical), $N_{or} \sim N_o$, so $C1^{ao} \sim C1^{no}$.

Therefore, in terms of computational cost, indexing with no orientation information is comparable to indexing with absolute orientation information since the reduction in data per bin is directly countered by an explicit orientation offset overhead. The complexity

(from above) is $(N_i N_m / N_{si}) N_v$, so data reduction results from subdividing into size bins, the orientation grouping does not enter into the complexity of the first search module. We compare this complexity to that of indexing using relative orientation information

$$\frac{C1^{ro}}{C1^{ao}} = \frac{((N_i N_m)^2 / (N_{si}^3 N_{or})) \cdot N_v}{((N_i N_m) / N_{si}) \cdot N_v}$$

$$\frac{C1^{ro}}{C1^{ao}} = \frac{N_i N_m}{N_{si}^2 N_{or}}$$

Therefore, absolute orientation indexing becomes more efficient when $N_i N_m > N_{si}^2 N_{or}$, which is the case for high data environments (complex models and high clutter, unsegmented imagery). Using the nominal values for N_{si} and N_{or} (18 and 36 respectively), $N_{si}^2 N_{or} \Rightarrow 10{,}000$. So when $N_i N_m > 10000$, that is any combination of the number of image vertices in a group times the number of model vertices per view is greater than about 10000, absolute orientation indexing will be more efficient than relative orientation indexing.

3D View Verification and 2D Transform Determination

This process is dominated by performing all pairwise associations on the

$$N_c = N_m M_i$$

$$= (N_m N_i) \ / \ (N_{si} N_{or})$$

candidate image vertices for each view hypothesis (see figure 2.7). Since for N elements there are (N * (N-1) / 2) pairs, the complexity is given by $(N_m M_i)^2$, so

$$C2 = (N_m N_i / N_{si} N_{or})^2$$

Here we can see explicity the role of Vertex Space in reducing complexity. Without Vertex Space, we would have complexity $(N_m N_i)$, so Vertex Space reduces N_m and N_i each by a factor of $(1 / N_{si} N_{or})$, so the reduction in complexity is $(1 / N_{si} N_{or})^2$.

End-to-end Feature Search

For low clutter (lc) environments where a clear peak for indexing into the view paramaters (θ_y, θ_x, θ_z) can be found, only a single view need be investigated. In this case, the end-to-end complexity is just

$$C^{lc} = C1^{ro} + C2$$

$$= ((N_i N_m)^2 / (N_{si}^3 N_{or})) N_v + (N_i N_m / N_{si} N_{or})^2$$

At the other extreme, very high clutter (hc) environments with respect to model complexity, useful peaks will be significantly degraded. In the worst case analysis, we assume that *all* views will need to be investigated. The resulting complexity is

$$C^{hc} = C1^{ao} + C2$$

$$= ((N_i N_m) / (N_{si} N_{or})) N_o N_v + (N_i N_m / N_{si} N_{or})^2 N_o N_v$$

$$= [((N_i N_m) / (N_{si} N_{or})) + (N_i N_m / N_{si} N_{or})^2] N_o N_v$$

where there are $N_v N_o$ view hypotheses which could be investigated. For this case, the second term clearly dominates, so we have

$$C^{hc} = N_o N_v (N_i N_m / N_{si} N_{or})^2$$

that is, the search is proportional to the number of views in the model database times $(N_i N_m)^2$ reduced by a $(1/N_{si} N_{or})^2$ factor, which acts to counter the effect of the multiple views. Essentially, we have a second order (in number of feature pairs) computation versus the direct model to image feature matching approach which is a third order problem. For large numbers of features, the computational savings is significant. The currently implemented approach does not assume any segmentation. That is, all image vertices are considered together as one large group. If grouping schemes are implemented (i.e., vertices grouped by proximity, perceptual significance..), then only subsets of the image vertices will be considered at one time, which can greatly reduce the computational burden.

After the feature based search is performed the final phase of the search involves mapping the entire target model into the image for image domain correspondence determination. While this is a relatively costly process, it is rarely required due to the powerful prescreening ability of the feature based search

STABLE FEATURE EXTRACTION

Feature based matching for ATR is criticaly dependent on the stability of extracted features. It is crucial that small changes in viewing conditions (i.e. geometry, lighting, occlusion..) do not result in substantial changes in the features extracted. In developing our high level feature matching search strategy, the emphasis was on search techniques and the task of extracting features from the image was deemphasized. During that stage of the program, only idealized vertex features were used. Simple, idealized, wire frame, polyhedral models were used which yielded unambiguous vertices that were simple to extract. The idealized image vertices were just instances of straight line edge

intersections. To extract these vertices we used straightforward standard techniques. Edges were extracted from images of simple models by calculating the gray level gradient magnitude and then thresholding. Since wire-framed models were used for rendering images, edge thicknesses were norminaly one pixel

Edges were mapped to contour structures by sequentialy storing pixels along the edge, which imposed pixel ordering on the extracted edges. Curvature was calculated along the contours and thresholded to find vertex sites. Curvature calculations required finding deriviates numericaly along the extracted contours, which was implemented by conlvolving with derivatives of the Gaussian. Convolution with Gaussians has been shown to be a stable, efficient and scale selectable technique for finding derivatives of image contours.

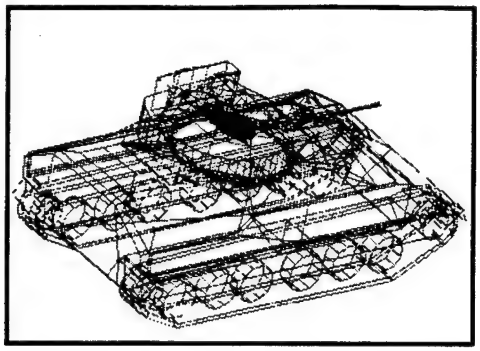
Our simple extraction techniques were adequate for development and evaluation of our search strategy, however, much more sophistication in feature extraction is required for real, complex models and imagery even though the search techniques still apply. Realistic treatment of complex models (usualy containing thousands of facets) requires image rendering using artificial light sources, simple wire frame renderings are not adequate. Consequently, variations in contrast must be dealt with. Further, the complex nature of the targets results in complicated topologies making contour tracing difficult and many important features, such as curvature extrema, can not effectively be treated as ideal vertices. Also, extracted edges often vary in width so that both edge breaks and thick edges are common occurences. A more subtle problem related to our ATR application is the need to extract only significant features that aid the recognition process and reject features that can not be reliably extracted and only confuse the ATR system. A complex target model consisting of thousands of facets can easily generate so many features that the search component could be overwhelmed. So one of the requirements of the feature extraction component is reduction of target signature to manageable levels.

To address these issues, we extended the feature extraction techniques developed for simple wire frame models. To deal with broken edges and thick edges we performed morphological processing on the extracted edge image. To fill holes we performed morphological dilation, which fills in small holes and thickens edges. The thick edges cause a problem when contours are traced, since the contours can wander within the thickened edges. Therefore, we followed dilation with thinning, which reduces the thickened edges to a nominal one pixel thickness (see figure 14 and 15).

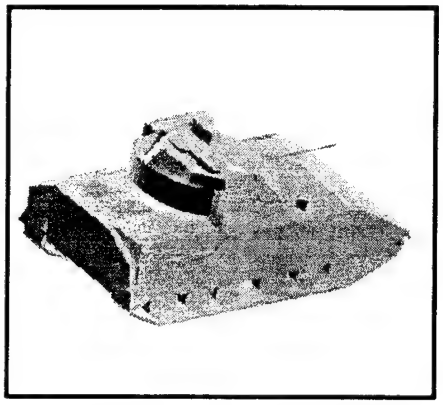
To deal with complex topology, which results in ambigous curve tracing when branches are encountered (see figure 15), all branch and end points are identified in the processed edge image (see figure 16a). Then the image is scanned until a branch or end point is encountered and all connecting contours are scanned until another branch or end point is found. The basic technique used to extract branch and end points is examination of all edge points using a circle centerd on the point of interest. The circle boundary is

Fig. 14) COMPLEX 3D TARGETS - rendering and feature extraction

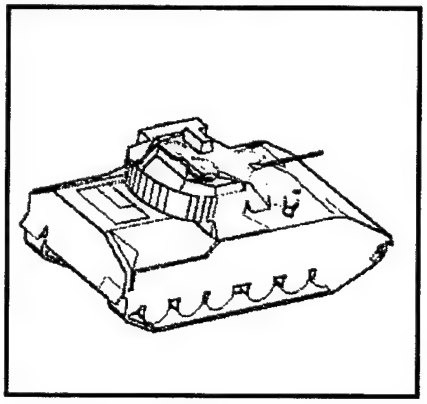
M2 Tank **(FRED file)**
1790 Facets



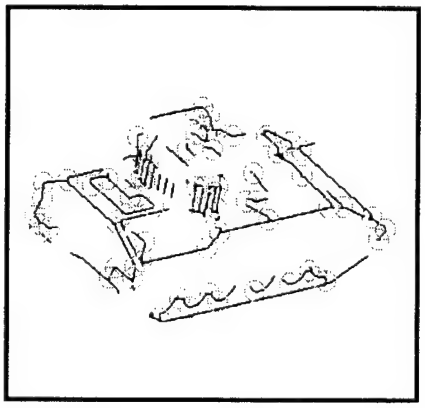
Wire Frame Rendering
(no hidden line removal)



Rendering with Lighting

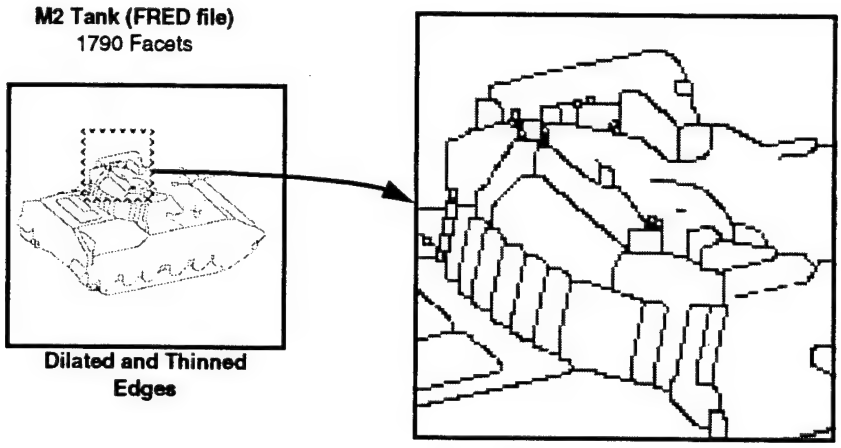


Processed Traced Contours and Vertices



Edge Extraction from Rendered Image

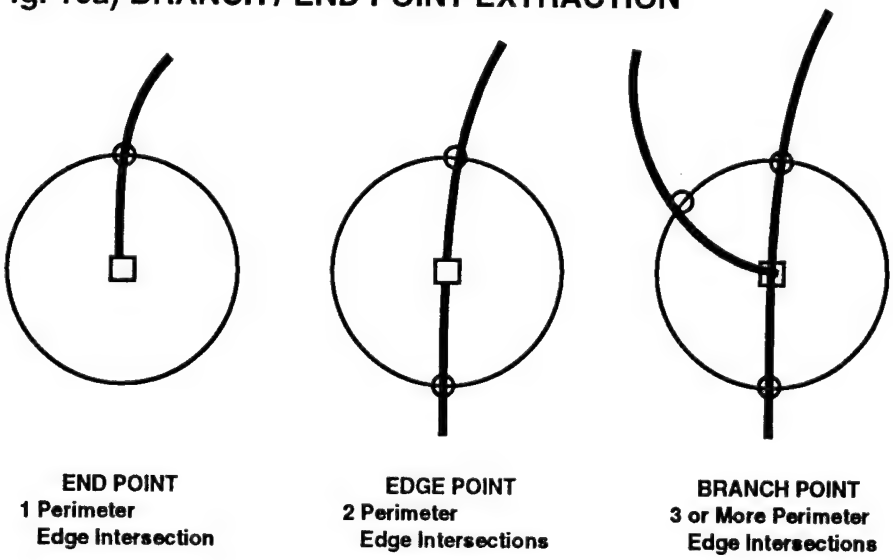
Fig. 15) COMPLEX 3D TARGETS - difficulty of curve tracing



scanned to determine the number of edges encountered (figure). One edge indicates the presence of an end point, two edges indicates a simple edge point and three or more encountered edges indicate a branch point.

Although the thinning process normaly results in single pixel edges, there are exceptions, especialy in the vicinity of branches. Therefore, non-edge to edge transitions are counted instead of edge pixels. Also, edges that approach each other to within a distance less than the circle radius used should not be considered in the edge count since the edges are not explicitly connected (small holes have already been filled by the dilation process). Therefore, we require edge pixels to be connected to the center pixel of the

Fig. 16a) BRANCH / END POINT EXTRACTION



circle through an edge segment to be considered. This process results in branch or end point regions, which consist of groups of connected pixels at branch or end locations. Each branch/end region is collapsed to a single pixel by replacing the region with the single branch/end point pixel with the most neighbors. This typicaly results in a subjectively optimum selection of branch/end points (figure 16b).

Any extracted contours less than a minimum length are rejected because it is likely they are not significant features that can be reliably extracted and will confuse the recognition process rather than aid it. The remaining contours are examined for curvature extrema (as described above). Vertices consist of the union of branch points and curvature extrema. A vertex thinning process, where groups of vertices within a small neighborhood are replaced by a single vertex as determined by a neighborhood filter (as described above), is used to eliminate redundant vertex detections. Vertex paramaters are found by calculating angles between lines from the vertex location to edge intersections with a circle centered on the vertex site.

This process results in what appear to be reasonable vertices (figure 17), however, when integrated into the ATR system, recognition performance is poor. Upon examination, it can be seen that the extracted vertices are very unstable with respect to small changes in viewing geometry due largely to artifacts introduced by the thinning operation (figure 15). We have developed one solution, which avoids thinning by optimal contour tracing of thick edges. A filter is used that stores at each pixel the number of edge pixels in its neighborhood. Contour tracing proceeds by progressing from one filter maximum to the next a set number of pixels away. If the filter output is considered to be a 3rd dimension of the dilated edge image, then our optimum contour following is

Fig. 16b) BRANCH / END POINT PROCESSING
Replace extracted branch/end region with single point surrounded by the most neighbors

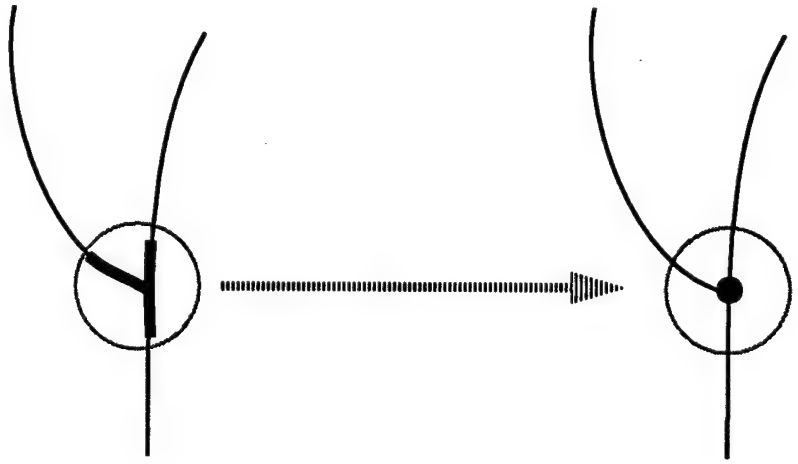
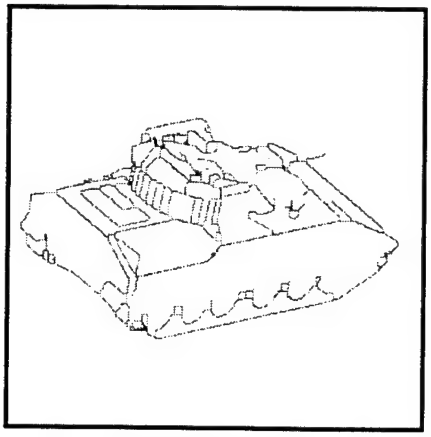


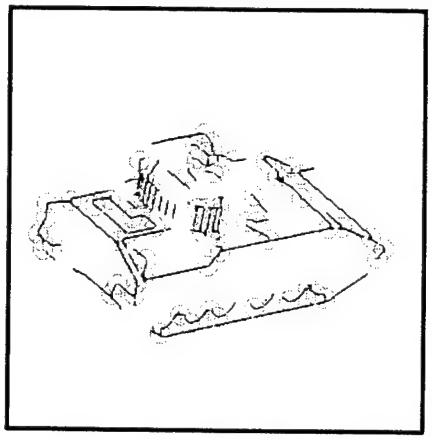
Fig. 17) COMPLEX 3D TARGETS - baseline feature extraction results

M2 Tank (FRED file)
1790 Facets

66 Vertex Locations
101 Vertices



Dilated and Thinned Edges



Processed Traced Contours and Vertices

analogous to following the ridge defined by the filter output local maxima.

While this approach seems promising and may provide a useful alternative to morphological thinning, we believe that vertex features can best be extracted using a direct local approach that does not require explicit edge and contour extraction. The topology of complex objects can be arbitrarily complicated, making any explicit edge following technique difficult and limited in applicability. Further, direct treatment of scale is required since scale can have a significant effect on the nature of vertex features extracted from imagery. In addition, calculated vertex paramaters should be stable with respect to the precise location of an extacted vertex.

To address these issues, we consider the geometric definition of curvature to motivate our approach.

If

F(x,y) is the image gray level at pixel site (x,y) and

C(x,y) is a contour pixel where the gradient of F(x,y) is a local maxima and

T(x,y) is the UNIT tangent of C at (x,y) and

N(x,y) is the UNIT normal to C at (x,y) (gradient direction)

so $\vec{N} = (\frac{F_x}{Q}, \frac{F_y}{Q})$ **where** $Q = \sqrt{F_x^2 + F_y^2}$ **, the magnitude gradient**

then Curvature is $K(x,y) = \left|\frac{d}{ds}\vec{T}(x,y)\right|$ **where** $\frac{d^\circ}{ds}$ **is the derivative of** $^\circ$

along the curve contour in the tangent direction

let $\vec{R} = \frac{d}{ds}\vec{T}(x, y)$ and the components of $\vec{R}$ be Rx and Ry,

such that $\vec{R} = (Rx, Ry)$, then $K = |\vec{R}|$.

So $Rx = (\frac{d}{dx}Tx \cdot Tx) + (\frac{d}{dy}Tx \cdot Ty)$ and $Ry = (\frac{d}{dx}Ty \cdot Tx) + (\frac{d}{dy}Ty \cdot Ty)$

we know from the properties of the unit tangent and unit gradient that

$\vec{T} \bullet \vec{N} = 0$ (from orthogonality); and $(Tx)^2 + (Ty)^2 = 1$

so $(Tx \cdot Nx) + (Ty \cdot Ny) = 0$

therefore $Tx = -(\frac{Ny}{Nx}) \cdot Ty$ and $(Tx)^2 = (\frac{Ny}{Nx})^2 \cdot (Ty)^2$

Then (with $\frac{dF}{d^\circ} = F_\circ$)

$$K = \frac{([(F_{yx} \cdot F_y) - (F_{yy} \cdot F_x)]^2 + [(F_{xy} \cdot F_x) - (F_{xx} \cdot F_y)]^2)^{1/2}}{Q^2}$$

(where $Q = \sqrt{F_x^2 + F_y^2}$)

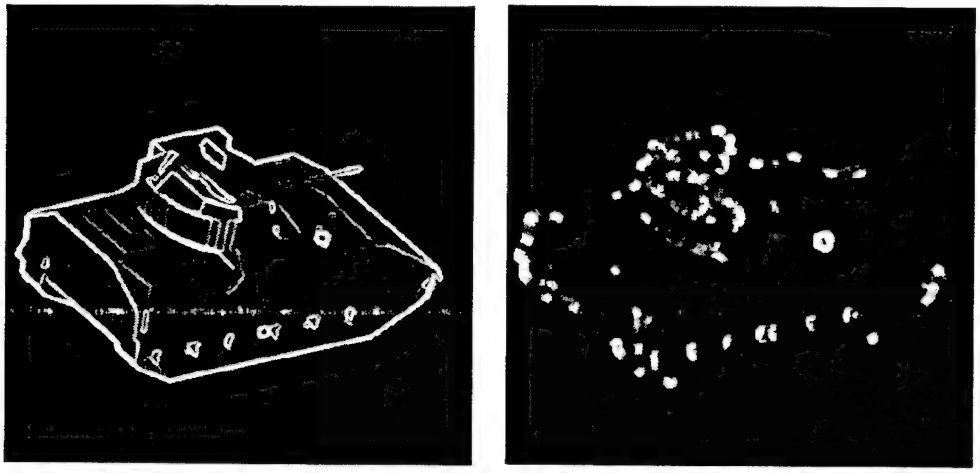
and finaly, after noting that F_{xy} and F_{yx} are equivalent, we have

$$\boxed{K = \frac{([(F_{xy} \cdot F_y) - (F_{yy} \cdot F_x)]^2 + [(F_{xy} \cdot F_x) - (F_{xx} \cdot F_y)]^2)^{1/2}}{F_x^2 + F_y^2}}$$

While this expression for curvature has a theoretical basis, there are many practical issues that must be considered. One issue is the well known numerical instablility of calculating high order spatial derivatives of images. Numerical differentiation tends to accentuate high frequency noise and the effect is more pronounced for succesively higher orders of differentiation. Also, in our application, we are most interested in small contour regions where branching or a high level of curvature occurs. These are regions where good numerical differences, which are required for the derivative calculations, are not well defined because image gray scale is changing quickly and in complex ways. There remain the questions of scale analysis and stable vertex paramater determination.

Therefore, although we have implemented the curvature calculation derived above (figure 18), we use it primarily as a point of departure. Rather than using numerical curvature explicitly, we use it as a conceptual guide to a more practical implementation. Conceptualy, geometric curvature is a measure of the change in orientation along an edge

Fig. 18) DIRECT CURVATURE CALCULATION



Gradient Magnitude of Tank Image

Extracted Curvature

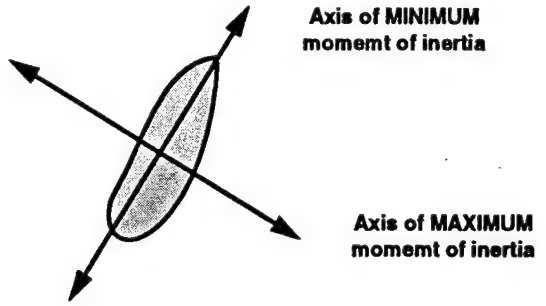
contour. Using this curvature concept, we proceed by finding local orientation at all potentialy interesting points (points where there is a non-zero gradient magnitude) in the image. The calculated orientation is analyzed by finding points where local orientations implicitly intersect. This is achieved by implicitly projecting a small distance in the direction of the local orientation and checking for the existence of a different orientation along the projection. For stable vertex detection, the implicit orientation change conditions described here must persist over some specified scale. Therefore, points of high curvature are found not only based on absolute changes in orientation but also by requiring the changes to occur along the direction of orientation, which is consistent with the spirit of geometric curvature.

To determine orientation at the lowest scale (highest sensitivity), we use the components of the gradient, where the x and y difference components are found using 1x3 (and 3x1) kernels. However, sensitivity is achieved at the cost of stability. To get more stable values at higher scale we would normaly increase the size of the kernels used to calculate the derivatives, however, this often results in increasing the ambiguity of the gradient calculation. To see one example of this, consider a strong, but narrow vertical step edge in an image (from left to right, light to dark to light). For a small kernel, the gray level difference in the x direction will be maximum at the optimal location where the gradient kernel straddles the step edge. If we increase the size of the kernel beyond the end of the step edge the x component value will decrease and the gradient output will be correspondingly degraded.

Alternatively, we calculate the principal axes of a region centered about each point of

Fig. 19) IMPLICIT EDGE ORIENTATION DETERMINATION

- **PRINCIPAL AXIS DIRECTION**
  (high stability, large kernel size)
  - principal axes are associated with maximum and minimum moments of inertia, and align with pre dominant directions



interest. Principal axes define dominant directions, or orientations in a region. In rigid body analysis, principal axes define axes of stable rotation for an object and will necessarily coincide with any axes of symmetry (see figure 19). To apply the principal axis calculation to imagery, mass density becomes the image gray scale value and we use only the two image dimensions, x and y. Principal axes are found by solving the 2 dimensional Eigenvector matrix equation $\underline{I}\vec{R} = \lambda\vec{R}$ where the two Eigenvector solutions are the principal axes and are necessarily orthogonal to each other. Here $\underline{I}$ is the moment of inertia matrix whose elements are

$$\underline{I} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

and $I_{jk} = \sum_{region} [F(x,y)\,(r^2\delta_{jk} - x_j x_k)]$ where $F(x,y)$ is the image gray level at location (x,y) and $r^2 = x^2 + y^2$. The matrix equation yields a secular equation, $|\underline{I} - \lambda\underline{\Lambda}| = 0$ where $\underline{\Lambda}$ is the identity matrix. So the Eigenvalues (moments of inertia about each axis) are

$$\lambda = \frac{(I_{xx} + I_{yy}) \pm \{ (I_{xx} + I_{yy})^2 - 4\,(I_{xx}I_{yy} - I_{xy}^2) \}^{1/2}}{2}$$

and $I_{xx}Rx + I_{xy}Ry = \lambda Rx$ or $Rx = \left(\frac{I_{xy}}{\lambda - I_{xx}}\right)Ry$, which defines the relations

between the Eigenvector components. To get actual component values, we impose the constraint that the Eigenvectors be unit length ($(\mathbf{Rx})^2 + (\mathbf{Ry})^2 = 1$). This procedure yields two Eigenvectors, one parallel to the orientation and one perpendicular. We choose the vector that corresponds to the minimum gray scale variance, which should be parallel to the local orientation A combination of the gray scale variance and ratio of the two Eigenvalues serves as a quality metric for each orientation value (small variances and large Eigenvalue ratios correspond to good, unambigous orientation values).

Having found the local orientations, we often wish to do some postprocessing, such as weighted average smoothing using orientation quality measures for the weighting. For any manipulations of this kind it is important to remove inherent orientation discontinuities that result from the discontinuous numerical values used to describe continous orientations (i.e., an orientation of 1 degree is relatively close to a 359 degreee orientation although the numerical difference is large).

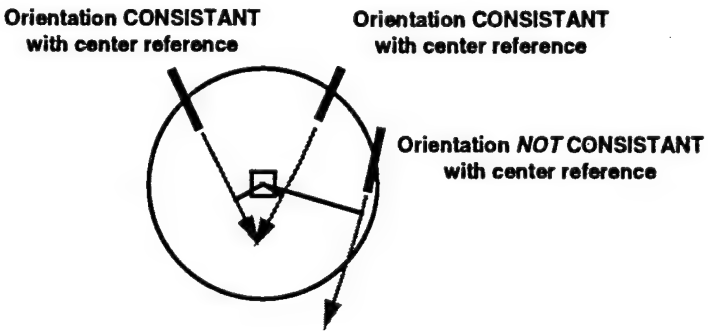
First, we note that, for our purposes, the direction sign of a vector is superfulous since the same orientation is desribed equivalently by a vector and its negative. To remove this redundency, we impose the constraint that all vectors be normalized to point to the first or second quadrant (0 to 180 degrees), which can be implemented by negating any vector whose y component is less than 0. The orientation angle of the vector is then doubled so the range is 0 degrees to 360 degrees. While the orientation values are not continuous, the corresponding transformed vector components are. Therefore, all operations are performed separately on the transformed vector components since vectors that define similar orientations will necessarily have similar transformed vector components and numerical discontinuites will be avoided. After processing, the inverse operations are performed on the resultant vector to find the true orientation.

The processed image orientation field is analyzed to extract generalized vertices which do not necessarily fit the idealized vertex model of two intersecting straight lines. A small circle centered on each point of interest (any point not inside a uniform gray level region) is used to examine the neighborhood. Any point on the circle whose orientation projects near enough to the center reference point is considered to be on the same implicit contour as the reference point, regardless of other region details (see figure 20). Therefore, two points can be assigned to the same implicit contour even if there exists contour "holes" between the points. If multiple adjacent points on the circle are found to be implicitly connected to the center reference point, they are replaced with the single pixel whose orientation projects closest to the center point, which effectively insures that implicit contours will be one pixel thick.

This process leaves one pixel on the circle for each implicit contour segment. These pixels are further examined for orientation consistency by stepping outwards in the direction of their orientation. If the priority is sensitivity, it is sufficient that the orientation at each step be consistent (project near enough) with the previous pixel. If, however,

Fig. 20) IPLICIT EDGE ORIENTATION ANALYSIS - CRITICAL POINT DETECTION

- **Find consistant surrounding orientations**
- **Compare consistant orientation values**



stability is required over sensitivity, the orientation at each step must be consistent with the original reference pixel, which is a more restrictive condition than that used to achieve higher sensitivity. In the former case, curved contours can be extracted while the latter case requires that contours be essentialy straight within the range of consideration. By varying the range of consideration we can control the scale and stability of extracted vertices so that very small features will not be extracted even if they appear to be highly curved.

Implicit contour segments that are orientationaly consistent over the required steps are used to determine whether or not a vertex is present. More than two implicit contour segments indicates a branch point and, therefore, also a vertex. If there are identicaly two contour segments, the difference in orientation at their ends is used as a measure of curvature. If the curvature is greater than the necessary threshold, a vertex is extracted and the calculated curvature value is used for the vertex size (figure 20). This approach results in stable vertex angle values with respect to small changes in the location of the vertex since absolute locations are not used to determine angles, only consistent orientation values in the neighborhood of the vertex location are used.

This vertex extraction process will typicaly result in groups of vertex pixels clustered together. For a stable vertex, the conditions that determine vertex existence must exist over a small region so there will necessarily be multiple vertex pixels output from the vertex extraction process. The raw vertex pixels (vertex locations) are processed so as to replace localized groups of pixels with a single optimum vertex pixel. First, a size filter is used that filters out all candidate vertex sites that are not part of a connected region of neighbor vertex sites greater than a minimum size threshold. The resulting vertex sites are dilated (thickened) so that close vertices are merged. The final processing stage collapses

vertex site regions down to a single vertex pixel by replacing the region with a single "best" pixel based on curvature and gradient strength.

RESULTS

Search Strategy Results

To test and evaluate our MBATR search strategy while de-emphasizing the role played by feature extraction, we have used synthetic imagery and rudimentary feature extraction techniques not appropriate for more complex imagery. We have generated simple 2D and 3D models and implemented interactive tools for manipulating and rendering images of the generated models.

One of the primary issues is ATR performance in the presence of clutter. To help analyze clutter effects, we can interactively add synthetic views of different objects to an image of an object we wish to recognize. To systematicly obtain more analytic clutter performance data, we randomly add clutter vertices to an image of a target to be recognized, in some cases with the target view unobstructed and clearly visible, in others, we obscure much of the target view and create large breaks in the contour of the object.

As one example, we have used a simple, idealized truck for a 3D target (see figure 21). We then generated a representative synthetic view of the truck, extracted the featuers, obscured many of the extracted features and added random clutter vertices to the image (see Figure 22). Our ATR algorithm was used to correctly identify candidate matching features (see figure 23). To further investigate performance in clutter, we systematicaly added clutter features one at a time to a portion of a target signature, measured the recognition elapsed time and stored the result in a file so that we can quantitatively study the relation between clutter and recognition time (see figure 24).

Stable Feature Extraction for Complex Targets

Using the implicit edge orientation approach optimized for sensitivity, described

Fig. 21) SIMPLE 3D TRUCK MODEL

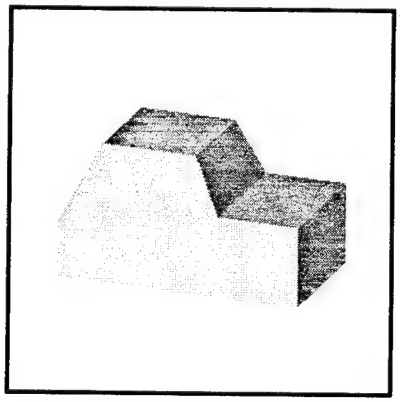


Fig. 22) SIMPLE 3D TRUCK MODEL MATCHING: INPUT IMAGE WITH 394 CLUTTER VERTICES

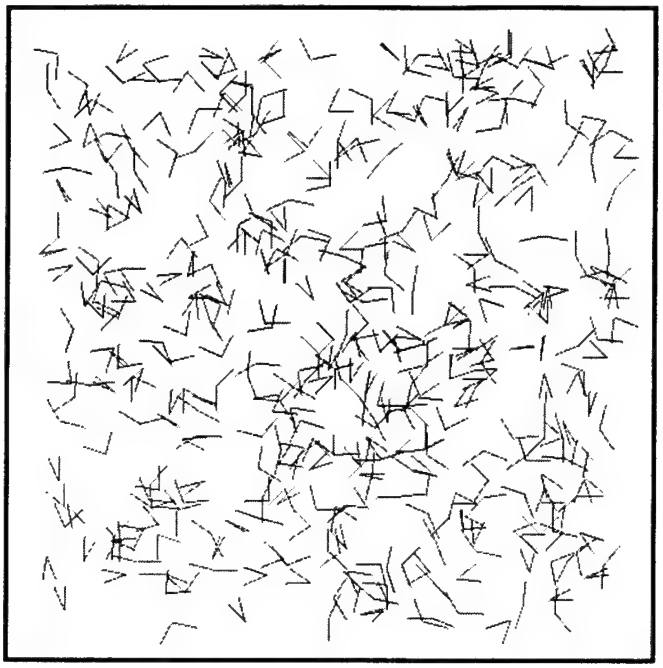
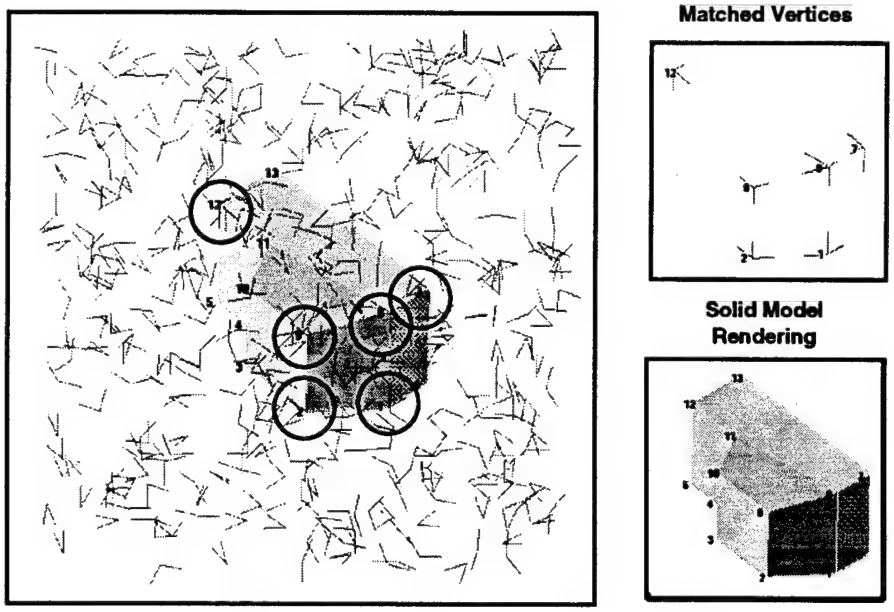


Fig. 23) SIMPLE 3D TRUCK MODEL MATCHING: RESULTS and GROUND TRUTH



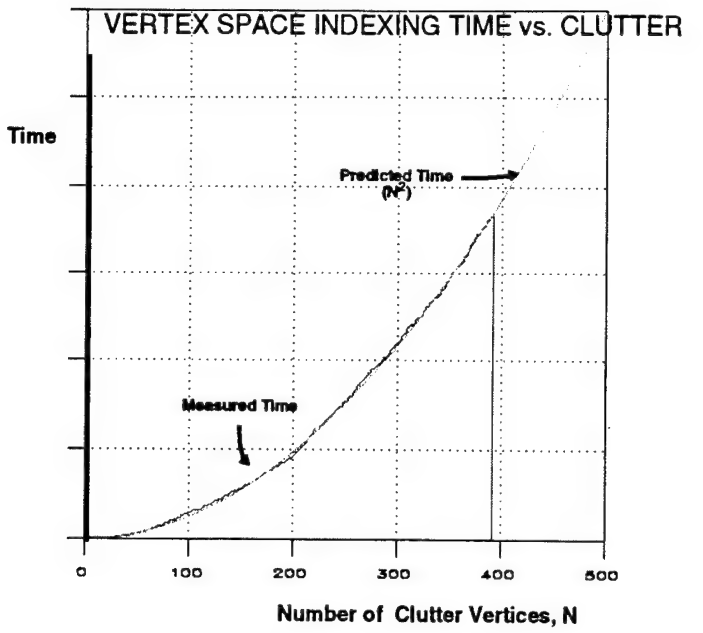


Fig. 24) VERTEX INDEXING CLUTTER TESTS - results
Indexing time versus clutter is measured by systematicaly adding random vertices to a degraded image of a target and recording the elampsed time for indexing. Note that the measured data is in close agreement with that predicted by the complexity analysis, namely, complexity should be proportionate to the square of the number of clutter (image) vertices.

above, we were able to extract useful features from real FLIR imagery, (see figure 25). However, this work must be considered preliminary since we did not proceed very far into the investigation with real imagery, as is evidenced by a sensitivity to parameter selection. It is a prime area for further investigation.

Recognition of Complex Targets

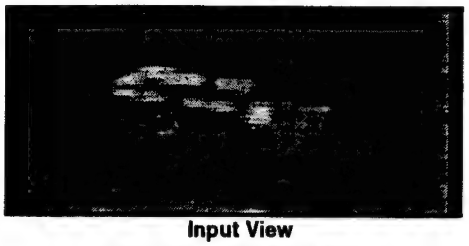
We used a model of an M2 tank consisting of 1790 facets to evaluate treatment of complex targets. Stable features were extracted from a typical synthetic target view and succesfuly used to find the best view stored in the target data base (see figure 26). The matched features are shown.

Unfortunately, at the end of the AMAPS program, we had not yet received from the appropriate organizations consistent sets of imagery and models, so we could not perform recognition tests of complex models using real imagery.

FUTURE WORK

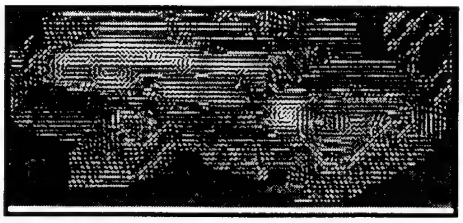
We plan to conduct further evaluation of the ATR capability already developed using data acquired (including additional models and imagery) after program completion.

Fig. 25) FEATURE EXTRACTION - real target, real IR imagery

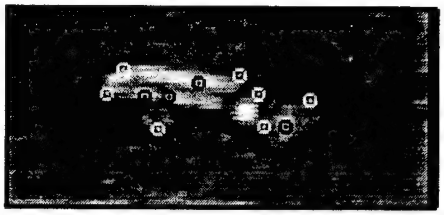


Imagery Provided by IMATH

Input View
First Generation FLIR Image



Extracted Orientation Using Principal Axes (Used for Implicit Edge Analysis)



Critical Points Extracted Using Orientation (note critical points over rear wheel based on implied edge)

Depending on the nature of problems identified by the evaluation process, algorithm refinements will likely involve a study of the effect of scale on recognition performance. It must first be determined what range of scale features will be stable over. Then a model database generation and search strategy that effectively address the issue of scale can be developed.

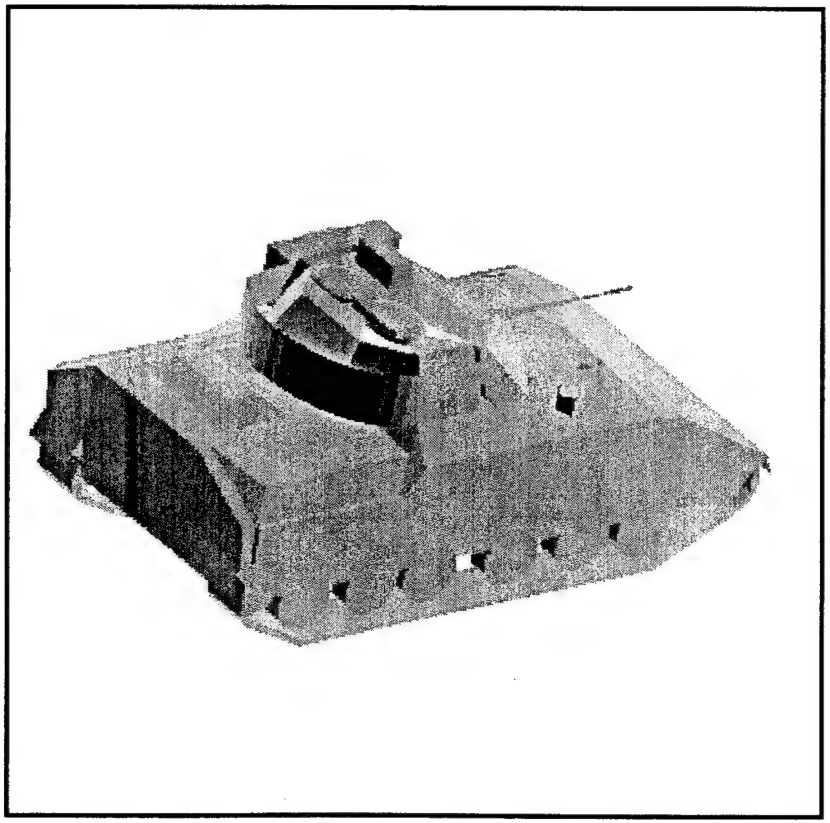
Another area of investigation involves modifying the Vertex Space representation. The current implementation maps vertex features, which are points of curvature maxima and branch points, to Vertex Space based on their angular size and orientation. This representation emphasizes discontinuity *points*, so that significant edges are only indirectly represented through their relations with other edges. For blob-like or unresolved signatures, vertex features may often be more ambiguous than the signature contours. Therefore, it may be appropriate to modify Vertex Space to map all points, not just discontinuous points, which would effectively accommodate *contours*. One way to do this would be to plot points based on their orientation and local change in orientation even if the change is zero (as in straight lines), instead of angular size as is done in the current Vertex Space implementation. To maintain a compact representation, points that are implicitly connected and lie in a single bin would have to be combined into a single entry which would necessarily require some scale information for the matching process.

It may also be appropriate to refine the model database generation procedure. Currently, sample views are taken uniformly about the target model. Alternatively, views could be taken adaptively based on stability of the signature. In viewing regions where the

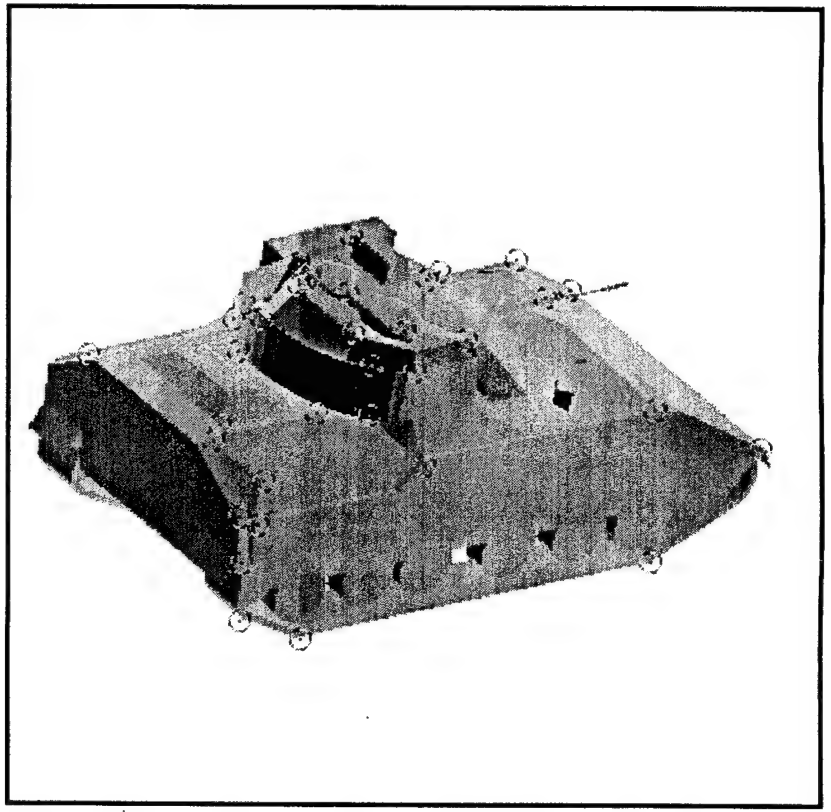
synthetically generated target signature is stable, less views are required and, conversely, more views are required where the target signature varies rapidly. An adaptive database generation process could better accommodate complex signature variations without increasing the amount of data to be stored and analyzed.

Significant improvement in efficiency can be achieved by implementing feature grouping techniques to reduce the size of image data that must be dealt with. A natural way to perform feature grouping is to extend the tangent lines associated with each extracted vertex until tangent lines from other vertices are encountered. Vertices connected in this way would define groups that would be examined using our current ATR techniques, but with significantly reduced data complexity.

Fig. 26a) COMPLEX 3D TARGETS - recognition results

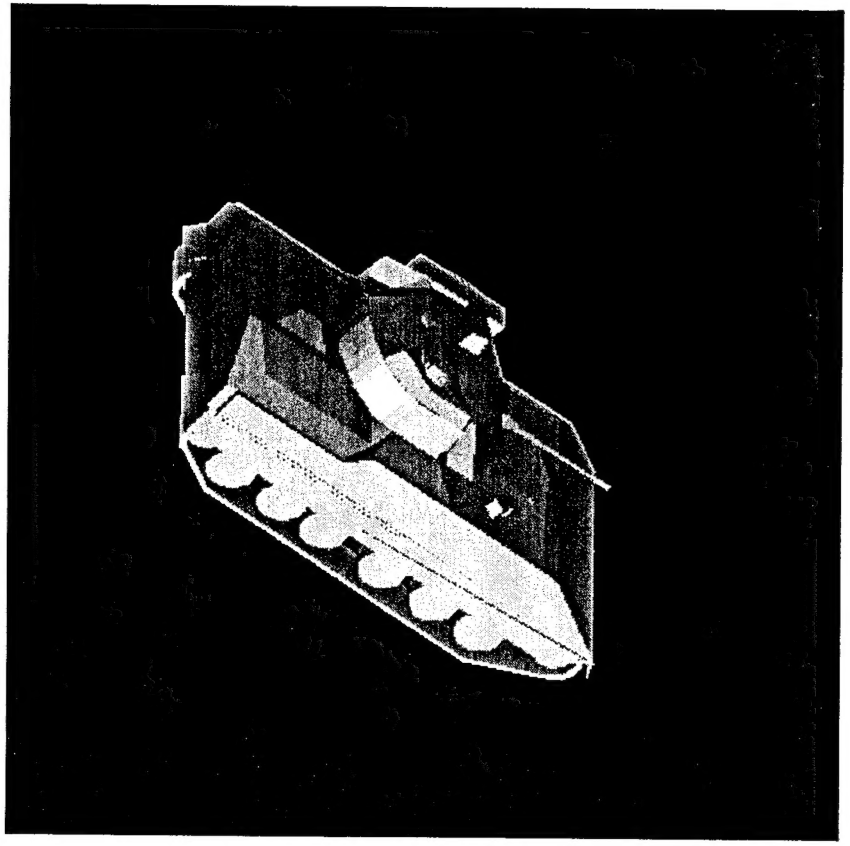


M2 Tank (FRED file) 1790 Facets
Input View (-40, 50, 50)

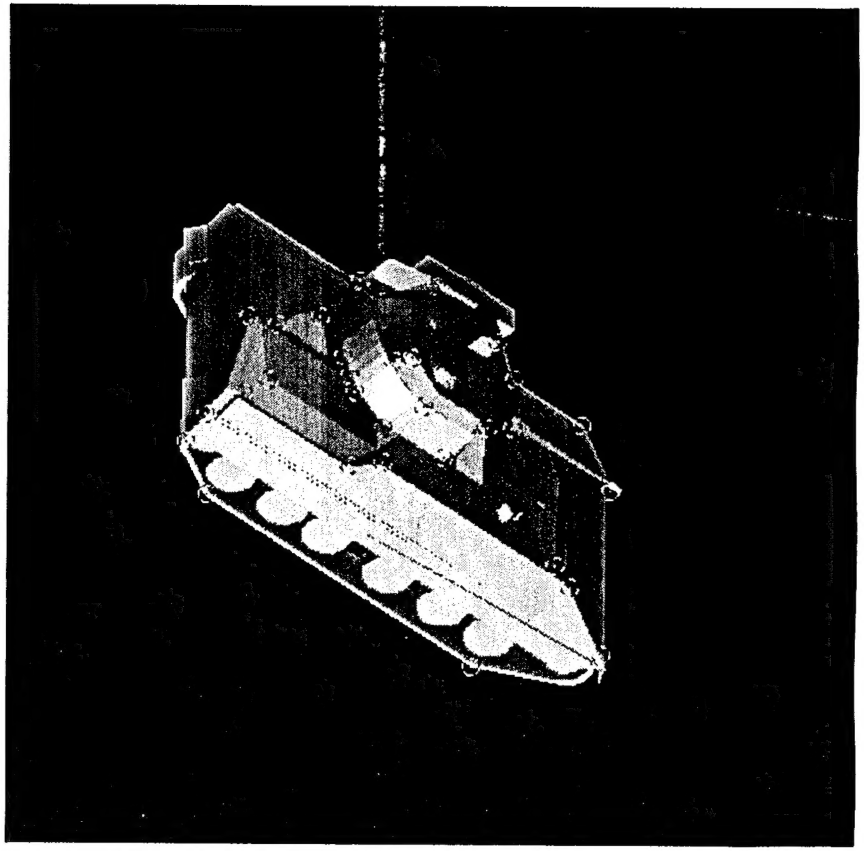


Input View
EXTRACTED VERTICES

Fig. 26b) COMPLEX 3D TARGETS - recognition results

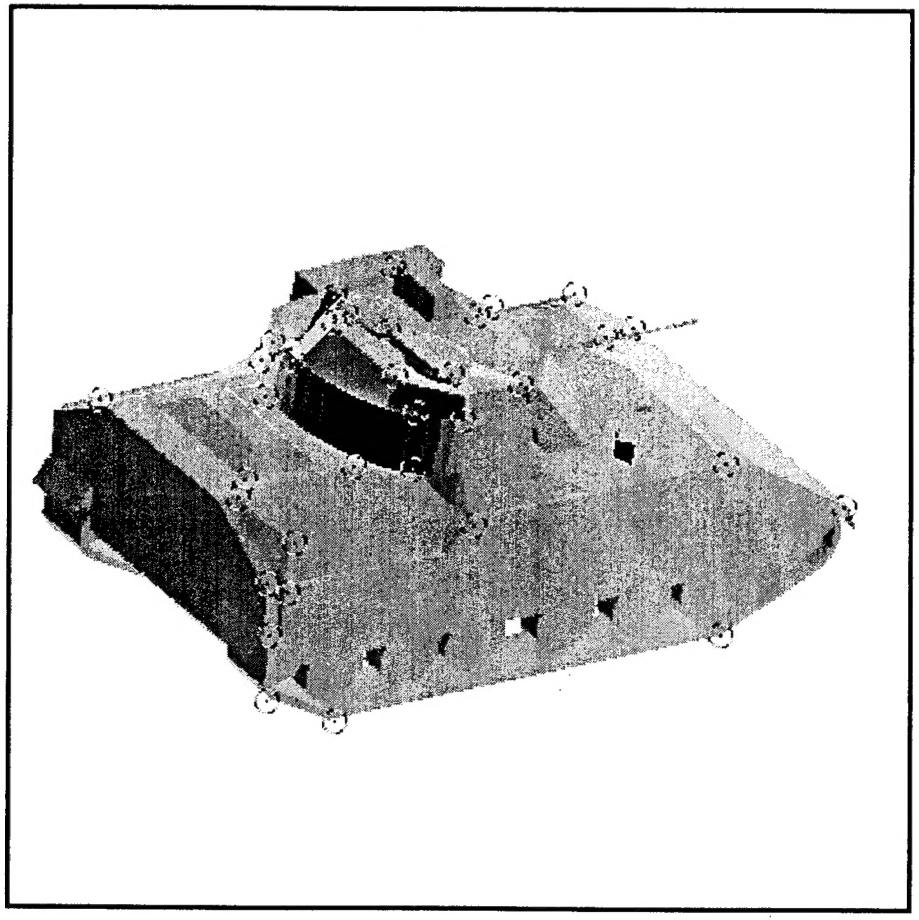


BEST MATCH VIEW
(-45, 42, 45)

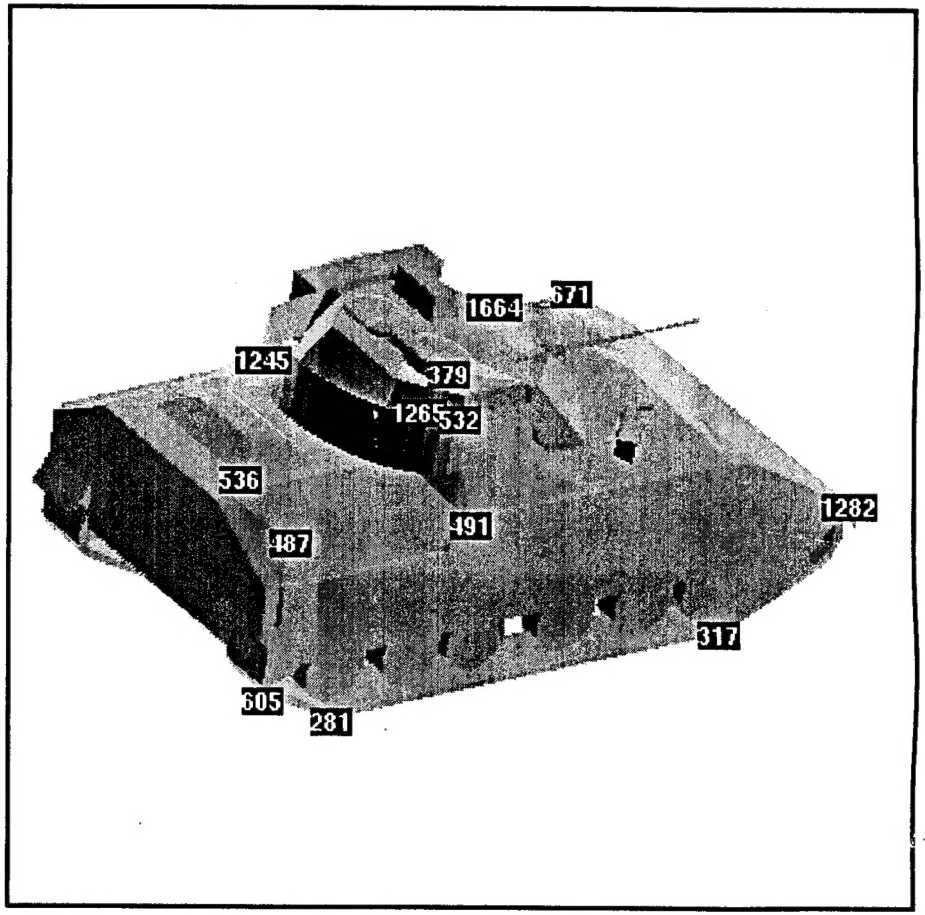


BEST MATCH VIEW
EXTRACTED VERTICES

Fig. 26c) COMPLEX 3D TARGETS - recognition results



Input View
EXTRACTED VERTICES



M2 Tank (FRED file) 1790 Facets
Input View (-40, 50, 50)
with matched features

OFFICE OF THE UNDER SECRETARY OF DEFENSE (ACQUISITION)
DEFENSE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22304-6145

IN REPLY REFER TO DTIC-OCC

SUBJECT: Distribution Statements on Technical Documents

TO: OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

1. Reference: DoD Directive 5230.24, Distribution Statements on Technical Documents, 18 Mar 87.

2. The Defense Technical Information Center received the enclosed report (referenced below) which is not marked in accordance with the above reference.

FINAL REPORT
N00014-92-C-0087
TITLE: AUTOMATED MISSILE AIM POINT SELECTION TECHNOLOGY

3. We request the appropriate distribution statement be assigned and the report returned to DTIC within 5 working days.

4. Approved distribution statements are listed on the reverse of this letter. If you have any questions regarding these statements, call DTIC's Cataloging Branch, (703) 274-6837.

FOR THE ADMINISTRATOR:

1 Encl

GOPALAKRISHNAN NAIR
Chief, Cataloging Branch

FL-171
Jul 93

DISTRIBUTION STATEMENT A:

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

DISTRIBUTION STATEMENT B:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT C:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT D:

DISTRIBUTION AUTHORIZED TO DOD AND U.S. DOD CONTRACTORS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT E:

DISTRIBUTION AUTHORIZED TO DOD COMPONENTS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT F:

FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date Below) or HIGHER DOD AUTHORITY.

DISTRIBUTION STATEMENT X:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DOD DIRECTIVE 5230.25, WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (Indicate date of determination). CONTROLLING DOD OFFICE IS (Indicate Controlling DoD Office).

The cited documents has been reviewed by competent authority and the following distribution statement is hereby authorized.

A

(Statement)

OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

(Controlling DoD Office Name)

(Reason)

(Controlling DoD Office Address, City, State, Zip)

DEBRA T. HUGHES
DEPUTY DIRECTOR
CORPORATE PROGRAMS OFFICE

19 SEP 1995

(Signature & Typed Name)

(Assigning Office)

(Date Statement Assigned)